

REMARKS

Claims 1-36 were pending. All were rejected. The Applicant has amended claims 1, 13 and 25 and added new claims 37-39. Therefore, claims 1-39 are presently pending. The Applicant requests further consideration and re-examination in view of the amendments above and remarks set forth below.

Objection to the Specification:

The Examiner objected to the specification under 35 U.S.C. § 112. Specifically, the Examiner indicated that in the detailed description of Fig. 3, item 342 is referred to as a value, as a plurality of values, and as a plurality of nodes. In addition, the Examiner indicated that items 342 and 344 are recited in the specification, but Fig. 3 shows items labeled 342-1, 344-1, 342-2, 344-2, and so on.

The Applicant has amended the written description so that the reference numerals in the text are used consistently with those in the drawings.

Rejections under 35 USC 103(a):

Claims 1, 2, 5, 7-14, 17, 19-26, 29, and 31-36 were rejected under 35 U.S.C. 103(a) as being unpatentable over Moreno et al. (U.S. Patent No. 5,758,051) in view of Bharadwaj (U.S. Patent No. 5,787,287). More particularly with respect to claim 1, the examiner stated that Moreno et al. show moving a second memory operation to an earlier position, closer to a first memory operation. The examiner further stated that Bharadwaj discloses features of claim 1 not disclosed by Moreno et al. and that it would have been obvious to modify the system of Moreno et al. with the features taught by Bharadwaj, for the purpose of determining whether reordering memory operations affects the meaning of the source code.

The Applicant overcomes the rejection by the above amendments to claims 1, 13 and 25. More particularly, the Applicant has amended claim 1 to include a step of "generating a summary for each memory operation in the graph representation that indicates for each location type used in the procedure the closest preceding memory operation to affect that location type...". This feature is supported by the Applicant's specification at least at page 18, line 29 to page 20, line 5. As is explained in the Applicant's specification, once the location type has been assigned for each load and store operation in the graph representation, the last operation that affected the same location type is determined. More particularly, as is explained in reference to Fig. 9A,

each node that represents a memory operation is examined to determine the location of the most-recent memory operation that affected that same location type. Thus, for each operation, the compiler generates a “summary” indicating the most-recent memory operation to affect each location type used in the procedure. As is shown in Fig. 9A, each operation is labeled with a node number and each node representing a memory operation includes a summary. For example, the summary for the node n5 is given as:

x.b = n3

x.a = n4

This summary indicates that for the node n5, the location type x.b was most-recently affected by node n3 and that the location type x.a was most-recently affected by node n4. This process continues for each node such that the summary lists all the location types affected by the procedure and for each such location type, the closest preceding node which affects that location type is indicated.

As is explained in the Applicant’s specification at page 20 lines 5-23, the summary information is used to move dependencies of operations by moving edges to memory operations that are at, or closer to the origin. This allows the compiler to reschedule the operations earlier to optimize speed performance while not altering the meaning of the program since the dependencies are respected.

Bharadwaj does describe a method for determining whether code motion or code reordering may be performed without altering the meaning of the source code. Bharadwaj at col. 6, lines 40-43. However, the disclosure of Bharadwaj does not suggest or disclose generating the summary as is now recited in amended claim 1.

More particularly, Bharadwaj discloses generating a path vector (PV) for a region represented by a flow control diagram. The PV has a length of $32 \times N$ bits and each path of a flow control diagram is mapped to a bit of the PV such that there is a one-to-one correspondence between bits in the PV and control flow paths.

Bharadwaj, at col. 3, line 50 to col. 4, line 3.

Then, each block within the region is characterized by a block path vector (BPV). The BPV has at least as many bits as paths in the region. A BPV indicates which paths pass through a given basic block based on the value of each bit.

Bharadwaj, col. 4, lines 26-36.

Then, dependency path vectors (DPV’s) are computed. The DPV of any two instructions, INSTR_A and INSTR_B is defined as zero if INSTR_B comes before

INSTR_A on any path and otherwise, the DPV is the bitwise logic AND of all BPVs that contain those two instructions. Bharadwaj, col. 4, lines 53-64. An exception occurs when a third instruction falling between those two instructions overwrites the variable, in which case, the DPV is computed using a different formula. Bharadwaj, col. 4, line 64 to col. 5, line 41.

If the compiler determines that it would be desirable to move an instruction to a new location to improve the speed of execution and the instruction is a writer of a variable, then all of the DPVs are recomputed between the instruction and all instructions that are readers of the variable. These DPVs are recomputed using the code as it would appear if the instruction had been moved. If there are differences between the post-code motion DPVs and the pre-code-motion DPVs, this indicates that the motion alters the meaning of the program. Bharadwaj, col. 6, line 55 to col. 7, line 4.

Thus, claim 1 as amended is not suggested or disclosed by Bharadwaj because Bharadwaj does not suggest or disclose generating a summary for each memory operation in the graph representation that indicates for each location type used in the procedure the closest preceding memory operation to affect that location type, as recited by claim 1. Rather, Bharadwaj teaches a completely different method for determining whether code motion alters the meaning of the program. A significant disadvantage of the technique of Bharadwaj is the need to compute multiple DPVs for a region of code and then for each proposed code motion the DPVs need to be recomputed and compared to the prior DPVs to see if the move is feasible. In contrast, the summary information of the present invention allows a compiler to move dependencies of operations by moving edges to memory operations that are at, or closer to, the origin without having to undergoing significant re-computation for each proposed move to determine whether the move alters the meaning of the code.

Moreno et al. do not suggest or disclose this feature of amended claim 1 either. Therefore, for at least this reason amended claim 1 is allowable over Bharadwaj and Moreno et al. taken singly or in combination. Claims 2, 5, and 7-12 are allowable at least because they are dependent from an allowable base claim 1.

Further, claims 13 and 25 have been amended similarly to claim 1. As such, ~~they, too, are allowable over Bharadwaj and Moreno et al. for these same reasons.~~ Claims 14, 17, 19-24, 26, 29, and 31-36 are allowable at least because they are dependent from an allowable base claim 13 or 25.

Claims 3, 4, 15, 16, 27 and 28 were rejected under 35 U.S.C. 103(a) as being unpatentable over Moreno et al. (U.S. Patent No. 5,758,051) and Bharadwaj (U.S. Patent No. 5,787,287) and further in view of Radigan (U.S. Patent No. 6,016,398). Claims 6, 18 and 30 were rejected under 35 U.S.C. 103(a) as being unpatentable over Moreno et al. (U.S. Patent No. 5,758,051) and Bharadwaj (U.S. Patent No. 5,787,287) and further in view of Radigan (U.S. Patent No. 6,151,704).

The Applicant overcomes the rejections by the above amendments to claims 1, 13 and 25. Thus, claims 3, 4, 6, 15, 16, 18, 27 and 28 are allowable at least because they are dependent from an allowable base claim 1, 13 or 25.

New claims 37-72:

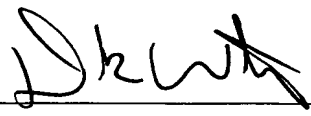
New claims 36-39 are added. Each recites that when no known preceding memory operation in the procedure affects a location type, the summary indicates the origin for that location type. This feature is supported by the Applicant's specification at least at page 19, lines 4-16. Claims 37, 38 and 39 are allowable at least because they are each dependent from an allowable base claim 1, 13 or 25.

Conclusion:

In view of the above, the Applicant submits that all of the pending claims are now allowable. Allowance at an early date would be greatly appreciated. Should any outstanding issues remain, the examiner is encouraged to contact the undersigned at (408) 293-9000 so that any such issues can be expeditiously resolved.

Respectfully Submitted,

Dated: April 1, 2004


Derek J. Westberg (Reg. No. 40,872)